VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015

Lukáš Farbjak

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Farbjak**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Tieto Czech s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

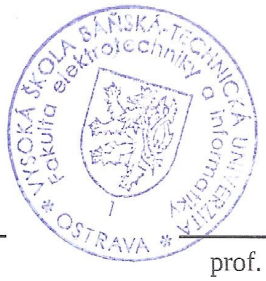
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Jan Platoš, Ph.D.**

Konzultant bakalářské práce: MSc. Szymon Skupień

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *22. dubna 2015*

.....
podpis studenta

Rád bych poděkoval společnosti Tieto Czech s.r.o. za možnost vykonání odborné praxe a všem jejím zaměstnancům, kteří se mnou spolupracovali.

Abstrakt

Cílem této bakalářské práce je popsat průběh odborné praxe, kterou jsem absolvoval ve firmě Tieto Czech s.r.o na pozici Software developer. Hlavní náplní praxe byl vývoj webových aplikací na platformě Java.

Začátek práce obsahuje představení společnosti a mé pracovní zařazení. Hlavní část se zabývá popisem zadaných úkolů a jejich řešením. V poslední části je popis uplatněných znalostí získaných v průběhu studia a také těch scházejících. Následuje zhodnocení dosažených výsledků a zkušeností, které mi praxe přinesla.

Klíčová slova: Odborná praxe, Tieto, Java, Spring framework

Abstract

The aim of this bachelor thesis is description of my professional practice, which I passed in the company Tieto Czech s.r.o. as a Software Developer. The main scope of practice was developing web applications in the Java platform.

The beginning of thesis contains presentation of the company and my assignment. The main part describes the tasks and their solution. At the last part is description of skills, which I used and also those, which I had not. The next is evaluation of the results and skills, which I have gained.

Keywords: Professional practice, Tieto, Java, Spring framework

Seznam použitých zkratek a symbolů

MVC	–	Model View Controller
SQL	–	Structured Query Language
POJO	–	Plain Old Java Objects
HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
XML	–	Extensible Markup Language
Java EE	–	Java Platform, Enterprise Edition
JSP	–	JavaServer Pages
LDAP	–	Lightweight Directory Access Protocol
Q&A software	–	Question and Answer software

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce popisuje průběh individuální odborné praxe, kterou jsem absolvoval ve firmě Tieto Czech s.r.o. Do společnosti jsem byl přijat na pozici Software Developer. Hlavním důvodem, proč jsem se rozhodl vykonávat odbornou praxi, bylo získání praktických zkušeností v oboru, které mi prozatím chyběly. Dalším důvodem bylo zlepšení dovedností ve vývoji informačních systémů a seznámení se s pracovní kulturou. O nabídce praxí ve společnosti Tieto jsem se dozvěděl z jejich internetových stránek. Pozici software developer jsem si vybral z důvodu, že se zajímám o programování a podobnou pozici bych chtěl vykonávat i v budoucím povolání.

2 Představení společnosti

Společnost Tieto byla založena v roce 1968 a má sídlo v Helsinkách. Čisté tržby společnosti činí 1,8 miliardy euro a zaměstnává 15 000 expertů ve více než dvaceti zemích světa. Tieto je největší dodavatel IT služeb pro soukromý a veřejný sektor ve Skandinávii. Do České republiky společnost Tieto vstoupila v roce 2001 a v současnosti má více než 2000 zaměstnanců a je jedním z největších zaměstnavatelů v oblasti IT služeb v České republice a největším v Moravskoslezském kraji. V roce 2004 společnost otevřela softwarové centrum v Ostravě.

2.1 Hlavní služby společnosti a její zaměření

2.1.1 Vývoj a správa aplikací

Vývoj a správa desktopových, webových a mobilních aplikací na platformě open source nebo Microsoft. Správa a optimalizace databází.

2.1.2 Portálová řešení

Tvorba podnikových portálů pro jednotlivé oblasti chodu organizací jako například projektové řízení a správa dokumentů.

2.1.3 Podnikové systémy

Podnikové informační systémy SAP. Nabízí procesně orientovaný a agilní přístup k řešení potřeb zákazníka.

2.1.4 Bussiness Intelligence

Nástroje a techniky, které umožňují rychlé a kvalitní rozhodování ve firmě.

2.1.5 Bussiness a IT transformace

Nabízí konzultační služby, které umožňují efektivní naplnění vize, strategií a požadavků pro společnosti prostřednictvím informačních technologií.

2.1.6 Aplikační outscourcing

Dodávka pracovníků, procesů, technologií a řešení pro řízení provozu IT infrastruktury nebo pro správu aplikací.

2.2 Pracovní prostředí a zařazení

Po kontaktování společnosti jsem nejprve absolvoval přijímací pohovor, kde si ověřili jak moje odborné znalosti programování v jazyce Java, tak úroveň angličtiny. Poté jsem nastoupil na vstupní školení, které trvalo dva dny. Zde jsem se dozvěděl informace o

pracovních podmínkách, struktuře firmy a také jsem podstoupil školení agilní metody Scrum určené pro vývojáře. Po nastoupení na nové pracovní místo jsem se seznámil s mým Line Managerem, konzultantem a také druhým studentem, který se mnou pracoval po celou dobu praxe. Byl jsem zařazen jako Software Developer na oddělení, které se věnuje open source technologiím a to především platformě Java. V průběhu praxe jsme spolupracovali s druhým studentem na stejných projektech, které náš konzultant jednou za čas zhodnotil a poskytl nám zpětnou vazbu.

3 Zadané úkoly

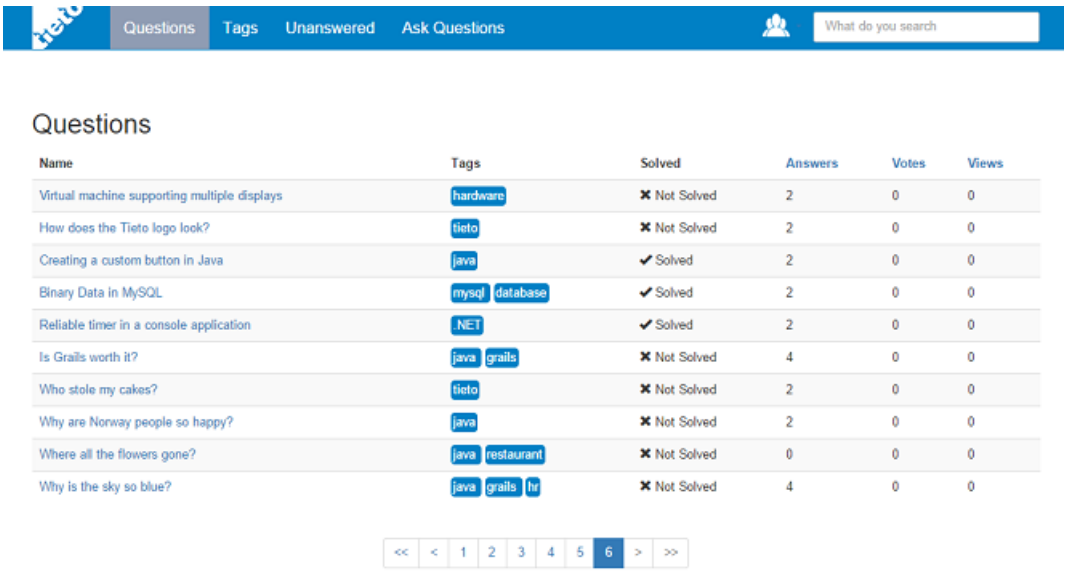
3.1 Q&A software - Tieto Ask

Prvním úkolem bylo navrhnout a implementovat webovou aplikaci s pracovním názvem Tieto Ask. Požadavkem bylo vytvořit takzvaný Q&A software, který by sloužil k zodpovězení dotazů v rámci společnosti Tieto. Například personální oddělení musí odpovídat mailem na stále stejné otázky, které jim zabírají čas. Proto vznikla myšlenka vytvořit aplikaci, která bude přístupná všem zaměstnancům společnosti, kde bude možnost vyhledat řešení jejich problémů. Pokud uživatel řešení nenajde, tak vytvoří novou otázku, na kterou budou ostatní odpovídat. Inspirací se stal portál stackoverflow.com, který se ovšem zaměřuje na pomoc programátorům. Tieto Ask je určena pro řešení problémů všeho druhu a cílovou skupinou jsou jak techničtí pracovníci, tak i ostatní zaměstnanci. Každá otázka má takzvané tagy, což jsou štítky označující druh otázky, které slouží k roztřídění otázek. Například pokud někdo bude mít problém s Javou, tak k otázce přidá tag "Java"a podobně. Každá otázka může obsahovat i více tagů. Uživatelé mají možnost otázky a odpovědí hodnotit a poté odpovědi s větším hodnocením jsou zobrazovány přednostně.

3.2 Czech Academy

Druhým úkolem bylo vytvoření informačního systému pro personální oddělení, který nahradí starší již používaný systém, který byl vytořen v Ruby On Rails. Systém slouží ke správě školících tréninků pro zaměstnance. Pracovníci personálního oddělení zde vytváří školení, kterému nastaví termíny konání a další parametry jako například cenu a pro jakou skupinu zaměstnanců je školení určeno. Zaměstnanci pak mají možnost se přes systém na tyto školení registrovat. Pracovníci z personálního oddělení pak získají přehled nad přihlášenými účastníky. Můžou účastníkům schvalovat jednotlivé školení, posílat hromadné zprávy, vytisknout prezence a podobně. Nenašel se žádný vývojář, který by udržoval starý systém a rozšiřoval jeho funkcionalitu, jelikož je ve firmě nedostatek zaměstnanců, kteří pracují s jazykem Ruby. Také zdrojový kód starého systému byl nepřehledný a bez dokumentace. Proto vznikl požadavek přepracovat systém do Javy, s kterou ve firmě pracuje více vývojářů. Cílem bylo vytvořit novou webovou aplikaci na platformě Java, která bude mít přehledný zdrojový kód a půjde snadno udržovat a dále rozšiřovat.

4 Vývoj aplikace TietoAsk



Obrázek 1: Ukázka aplikace Tieto Ask

Aplikaci Tieto Ask jsme vyvíjeli od úplného začátku až po nasazení na server. V následujících podkapitolách popíšu nejdůležitější části, které nás provázeli vývojem aplikace.

4.1 Specifikace požadavků

Nejprve jsme se od našeho konzultanta dozvěděli informace o systému, který je potřeba vytvořit a sepsali stručnou specifikaci požadavků a funkcionality. Dále byl vytvořen návrh vzhledu aplikace. Základní požadovaná funkcionalita:

- Webová aplikace na platformě Java
- Vkládání otázek a odpovědí
- Přihlášení uživatelů pomocí firemního LDAP
- Hodnocení otázek a odpovědí
- Možnost zobrazení otázek rozdělených podle takzvaných tagů
- Vyhledávání otázek a odpovědí
- Odpovědi s vyšším hodnocením se zobrazují přednostně
- Možnost označit odpověď, která vyřešila problém
- Editace odpovědí a otázek jejich autorem

4.2 Založení aplikace

Pro tvorbu aplikace jsme využívali vývojové prostředí Netbeans. Aplikace byla založena jako Maven projekt. Apache Maven je nástroj používaný pro vývoj Java aplikací, který umožňuje definovat externí závislosti projektu (Artefakty) a usnadňuje práci při sestavení aplikace. V kořenovém adresáři každého Maven projektu se nachází konfigurační soubor pom.xml. V tomto souboru definujeme všechny artefakty pomocí parametrů jako jsou groupId, artefactId, version. Maven pak sám dodá potřebné knihovny ze svého repozitáře a sestaví celý projekt. Příklad konfigurace pom.xml:

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.tieto</groupId>
    < artifactId >TietoAsk</ artifactId >
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            < artifactId >spring-core</artifactId>
            <version>4.1.4.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            < artifactId >spring-web</artifactId>
            <version>4.1.4.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            < artifactId >spring-webmvc</artifactId>
            <version>4.1.4.RELEASE</version>
        </dependency>
    </dependencies>
</project>
```

Výpis 1: Ukázka konfigurace souboru pom.xml

V příkladu jsem ukázal konfiguraci Maven projektu, který obsahuje závislosti na Spring framework verze 4.1.4, kterou jsme použili také v naší aplikaci. V průběhu vývoje aplikace jsme postupně přidávali další závislosti podle potřeby. Aplikace byla rozdělena do vrstev podle třívrstvé architektury, proto byl hlavní projekt rozdělen na další tři podprojekty, které reprezentovaly datovou, aplikační a prezentační vrstvu. Pro sdílení zdrojového kódu aplikace jsme využívali nástroj Git.

4.3 Vytvoření datového modelu

K vytvoření datového modelu jsme využili Hibernate framework. Z jeho pomocí jsme mapovali Java třídy na entity v relační databázi. Každou třídu, kterou jsme potřebovali

mapovat do databáze, jsme označili anotací. Hibernate poté dokáže automaticky vytvořit datový model v příslušné databázi podle konfigurace. Příklad mapování entity s využitím anotací v Java třídě:

```
@Entity
@Table(name="Question")
public class Question implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    private String title ;
    @Column(columnDefinition="TEXT")
    private String description;

    private Boolean solved = false;

    private Integer views = 0;

    @Temporal(javax.persistence.TemporalType.DATE)
    private Date dateCreated;

}
```

Výpis 2: Mapování entity pomocí Hibernate

Podle uvedeného příkladu Hibernate vytvoří v databázi tabulku s názvem Question, která bude mít sloupce id, title, description, solved a dataCreated. Atribut id bude automaticky generovaným primárním klíčem a všechny atributy budou mít požadované datové typy. Můžeme vidět, že anotace jsou použity nad názvem třídy a také u atributů.

Pomocí anotací také vytváříme vztahy mezi relacemi v databázi. V následujícím příkladu jsem uvedl, jak vytvořit dvě tabulky s kardinalitou 1:N, kdy jedna otázka bude mít více odpovědí:

```
@Entity
@Table(name="Question")
public class Question implements Serializable {

  @OneToMany(mappedBy = "question", fetch = FetchType.LAZY)
   private List<Answer> answers;
}

@Entity
@Table(name="Answer")
public class Answer implements Serializable {

  @ManyToOne()
   @JoinColumn(name = "question_id")
  private Question question;
}
```

Výpis 3: Mapování vztahů mezi relacemi pomocí Hibernate

Podle uvedeného příkladu budou v databázi vytvořeny tabulky Question a Answer se vztahem 1:N a tabulka Answer bude obsahovat cizí klíč question_id. Podobným způsobem lze vytvořit také vztahy M:N a 1:1.

Pro správnou funkci Hibernate je nutné, aby aplikace měla daný JDBC driver podle použité databáze. V našem případě jsme používali databázi PostgreSQL. Dále je potřeba nastavit adresu databáze pomocí konfiguračního souboru ve formátu xml.

V našem projektu jsme využívali výhody snadné modifikace datového modelu, kterou Hibernate nabízí. V naší aplikaci docházelo k častým změnám atributů a ke vzniku nových entit, jelikož aplikace vznikala od úplných základů a nové požadavky postupně přibývaly při vývoji aplikace.

4.4 Spring Framework v naší aplikaci

Největší část naší aplikace využívá právě Spring framework. Na začátků bych chtěl zmínit, že Spring je opravdu velmi rozsáhlý framework složený z několika částí, které jsou ještě dále rozděleny do modulů. My jsme využívali kromě části Core Container (Jádro frameworku) především část Web, která obsahuje Spring Web MVC. Spring Web MVC je implementací konceptu Model-View-Controller (MVC) pro webové aplikace. V současnosti jde o velice oblíbený koncept vývoje webových aplikací na všech platformách a nabízí přehledné psaní kódu, proto jsme se ho rozhodli využít i v naší aplikaci.

Základem Spring MVC je jeden hlavní servlet s názvem DispatcherServlet, kterému se také říká Front Controller. Ten zpracovává veškeré HTTP requesty, které přes něj putují k ostatním controllerům. Model ve Springu představuje libovolnou třídu, která obsahuje data. V našem případě jsme většinou využívali třídy ModelMap, které můžeme přidávat libovolný počet atributů s daty. Controller je ve springu třída, která je namapovaná na určitou URL a zpracovává HTTP requesty. Controller vykoná požadované akce, naplní data do modelu a zobrazí určitý view. Typická webová aplikace obsahuje více tříd Controller. View ve Spring MVC představuje nejčastěji JSP stránku.

Naše celá webová aplikace byla založená na MVC. Snažili jsme se dodržovat konvenci, že controller má jméno podle entity, se kterou pracuje. Takže Controller, který zobrazoval otázky, se jmenoval QuestionController a podobně. Dále jsme se snažili dodržet aby URL, kterou controller přijímal, byla složena z jeho názvu a za lomítkem byla upřesněna jeho akce. Takže controller, který spouštěl akci pro vytvoření nové odpovědi, přijímal URL ve tvaru "answer/create". Dále můžou být součástí URL také GET parametry, které Controller přijímá.

Chtěl bych zdůraznit, že MVC by měl pracovat na prezentační vrstvě a tuhle konvenci jsme dodržovali i u naší aplikace. Controller tedy nepracoval přímo z databází, ale do modelu přidával pouze data, které získával od metod ze servisní třídy. V následujícím příkladu uvedu, jak vypadá Controller, který zajistí zobrazení určité otázky:

```
@Controller
public class QuestionController {

    @Autowired
    private QuestionService questionService;
```

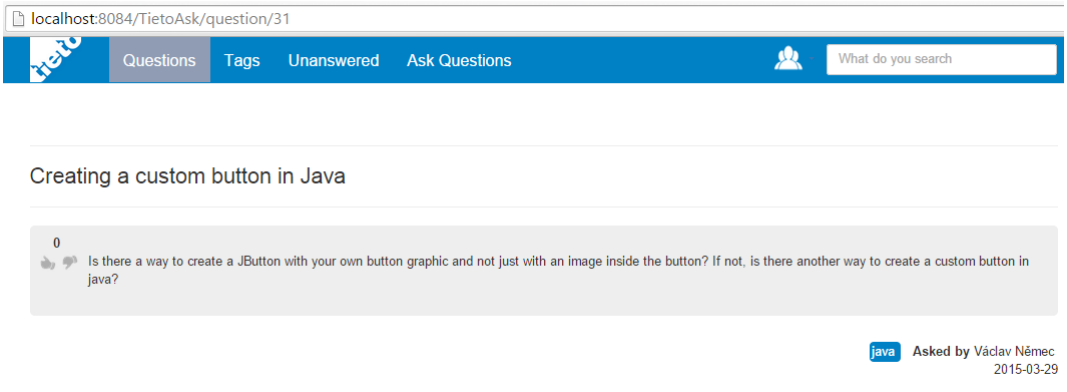
```
    @RequestMapping(value = "/question/{id}", method = RequestMethod.GET)
    public String  show(ModelMap model, @PathVariable int id) {

        Question question = questionService.findById(id);
        model.addAttribute("question",  question);

        return "question/show";
    }
}
```

Výpis 4: Controller s metodou pro zobrazení otázky

V ukázce můžeme vidět třídu QuestionController, která je označena anotací "@Controller". Tento Controller obsahuje metodu show a v anotaci "@RequestMapping"má uvedenou URL, kterou bude přijímat a taky parametr method, který uvádí metodu HTTP požadavku, v našem případě GET. Metoda obsahuje parametr model, který Controller naplní daty a parametr id s anotací "@PathVariable". Anotace "@PathVariable"ve Springu znamená, že v URL se nachází atribut, který je umístěn ve složených závorkách. Tím je umožněno předávání proměnných pomocí HTTP požadavků. V našem případě předáváme id otázky. Uživatel klikne na odkaz otázky s určitým id a Controller spustí metodu servísní třídy, která načte požadovanou otázku. Můžeme si všimnout, že nikde nevytváříme instanci servisní třídy a přímo s ní pracujeme. Je to kvůli odstranění těsných vazeb ve Spring, kdy se třída označí anotací "@Autowired"a framework podle potřeby vytvoří instanci za nás. Poté je objekt otázky předán do modelu. V našem případě model představuje mapu, které se předá název klíče a příslušný objekt. Podle názvu klíče, pak objekt najdeme v JSP stránce. Metoda vrací řetězec "question/show", který označuje adresu k příslušné view. Skutečná adresa umístění JSP stránky je delší, ale DispatcherServlet k řetězci přidá nastavenou předponu a příponu.



Obrázek 2: Výsledná view, která zobrazí otázku

4.5 Práce s databází

Pro prácí s databází jsme využívali JPA Repositories, které jsou součástí modulu Spring JPA, který je umístěn v Spring Data. JPA repositories jsou rozhraní, které poskytují CRUD operace pro spravované entity. Ukázka JPA repository pro entitu Question v naší aplikaci:

```
public interface QuestionRepository extends JpaRepository<Question, Integer> {

    Question findById(Integer id);

    List<Question> findBySolved(boolean solved);

    @Query("SELECT q FROM Question AS q LEFT JOIN q.answers qa GROUP BY q
        ORDER BY COUNT(qa) ASC")
    public Page<Question> findAllSortedByAnswers();

}
```

Výpis 5: JPA repository pro entitu Question

V ukázce vidíme rozhraní QuestionRepository, které dědí z JpaRepository. Jako parametr se pro JpaRepository dosadí typ Question, který představuje entitu a Integer, který je typ atributu klíče dané entity. Vytváření dotazů je velmi jednoduché. První metoda findById vyjadřuje, že bude vytvořen dotaz nad databází, který vrátí otázku vyhledanou podle id. Dále nemusíme nic víc definovat a dotaz Spring Data JPA automaticky vygeneruje. Podle definice druhé metody findBySolved bude vytvořen dotaz, který vrátí otázky vyhledané podle parametru solved. Jpa Repositories obsahují ještě daleko více klíčových slov pro generování dotazů. Jejich celý seznam a použití je uveden v oficiální dokumentaci pro Spring Data JPA. Pro používání složitějších dotazů je možnost vytvořit vlastní dotaz. U třetí metody findAllSortedByAnswers je ukázáno, jak takový dotaz napsat. Nad metodou se použije anotace "@Query"a do závorky se napíše dotaz, který je velmi podobný jazyku SQL. Některé jednodušší metody jako save pro uložení otázky do databáze, se nemusí ani definovat a naše repository je umí automaticky, protože je zdědila z obecného rozhraní JpaRepository.

4.6 Servisní třídy

V naší aplikaci jsme vytvořili třídy, které slouží pro vykonávání operací na servisní (aplikační) vrstvě. Servisní třídy tvoří prostředníka mezi Repositories v datové vrstvě a Controllers v prezentační vrstvě. Pokud bylo potřeba například uložit otázku do databáze, tak controller zavolal metodu příslušné servisní třídy a ta zase využila metod z repository k poskládání výsledné operace. Servisní metody můžou obsahovat množinu operací a využívat k tomu více tříd, narozdíl od metody rozhraní Repository, které vykoná pouze jednu určitou CRUD operaci. Příklad servisní třídy v naší aplikaci:

```
@Service
public class AnswerService {
```

```
    @Autowired
    private AnswerRepository answerRepository;
    @Autowired
    private QuestionRepository questionRepository;
    @Autowired
    private PersonRepository personRepository;

      @Transactional
    public void save(Answer answer, Integer questionId, String login)  {

        Person person = personRepository.findByLogin(login);
        if  (person != null)  {

            Question question = questionRepository.findOne(questionId);
            answer.setAuthor(person);
            answer.setQuestion(question);
            answer.setDateCreated(new Date(System.currentTimeMillis()));

            answerRepository.save(answer);
        }
    }
}
```

Výpis 6: AnswerService s metodou pro uložení otázky

V ukázce vidíme třídu AnswerService s anotací "@Service", která ve Spring framework znamená, že se jedná o komponentu servisní vrstvy. Metoda save slouží k uložení odpovědi. Servisní metoda odpovědi nastaví autora, otázku na kterou se odpovídá a datum vytvoření. Anotace "@Transactional"nad metodou zajistí, že budou všechny operace provedeny v jedné transakci. Pokud chceme, aby všechny metody servisní třídy byly vykonány jako transakce, tak zapíšeme anotaci "@Transactional"nad třídou.

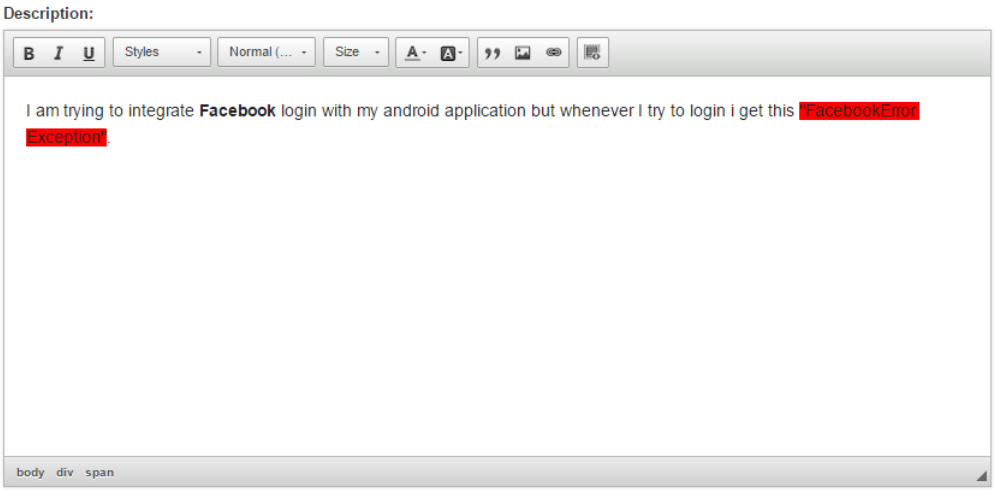
4.7 Tvorba šablony webové aplikace

Při volbě vzhledu aplikace jsme zvažovali několik možností. Jedna z nich byla stáhnout hotovou šablonu, ale bylo těžké najít takovou, která by byla zdarma a přesně odpovídala naším potřebám. Další možností bylo vytvořit novou šablonu od základů, ale tahle možnost by zabrala příliš moc času a navíc nikdo z nás nebyl nadaný webový designer. Nakonec jsme využili nástroje Twitter Bootstrap, který nabízí mnoho interaktivních elementů vytvořených pomocí HTML, CSS a Javascriptu. Tyto elementy lze velmi snadno vložit do jakékoliv webové aplikace. Stačí pouze stahnout potřebné soubory (CSS a Javascript), které se naimportují do stránky. Původním HTML elementům, které chceme nahradit, obvykle stačí pouze nastavit požadovaná CSS třída. Pomocí CSS je možnost také elementy dále upravovat podle vlastní potřeby. Tímto způsobem se dá jednoduše vytvořit velmi efektivní vzhled stránek s responzivním designem.

Dále jsme využili framework Apache tiles, který nám umožnil rozdělit stránku do několika částí a tím vytvořit výslednou šablonu stránky.

4.8 Javascriptové komponenty

V naší aplikaci jsme využívali několik Javascriptových nebo Jquery komponent podle potřeby. Většinou bylo výhodnější stáhnout hotovou komponentu, která byla zdarma. Vývoj vlasních komponent by zabral spoustu času a výsledek by pravděpodobně nebyl ani tak kvalitní. Jako jednu ze zásadních bych zmínil CKEditor. Jde o opensource WYSIWYG textový editor, který jsme v naší aplikaci využili pro vkládání popisu otázek a odpovědí. Uživatel tím získal možnost naformátovat vložený text podobně jako třeba u aplikace word.



Obrázek 3: Ukázka použití CKEditor

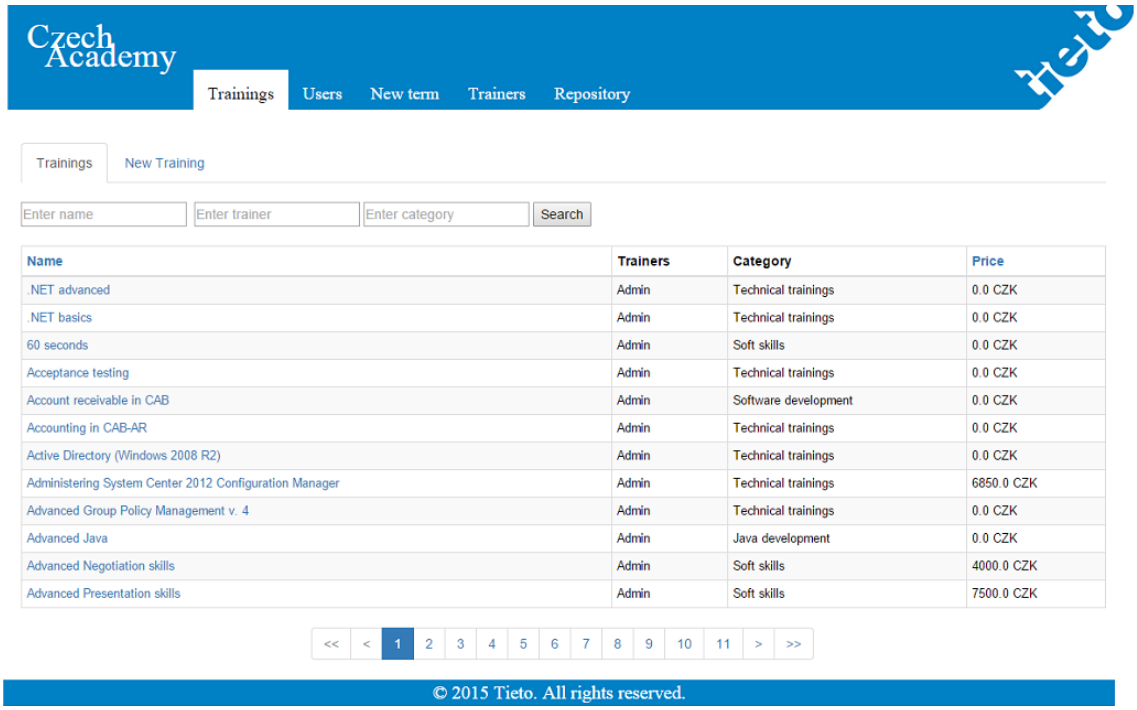
4.9 Nasazení aplikace

V průběhu praxe jsme dostali možnost provést nasazení naší aplikace na server. Do té doby jsme aplikaci zkoušeli pouze na vlastních počítačích. Získali jsme přihlašovací údaje k serveru, kde byl nainstalován systém Linux Ubuntu. Pro připojení jsme využívali SSH protokol. Na server jsme nainstalovali databázi PostgreSQL a webový server Apache Tomcat, který spouštěl samotnou aplikaci.

4.10 Shrnutí vývoje aplikace

Aplikaci TietoAsk jsme věnovali nepřetržitě prvních 25 dní naší praxe. V době psaní této práce už byla většina funkcí dokončená a probíhaly pouze úpravy detailů. Paralelně s vývojem aplikace TietoAsk se naskytla možnost pracovat na další aplikaci Czech Academy, které jsme věnovali v další části praxe už většinu času.

5 Vývoj Czech Academy



Obrázek 4: Ukázka systému Czech Academy

K vývoji Czech Academy jsme využívali podobných technik jako u předchozího Tieto Ask. Proto už nebudu podrobně popisovat práci na jednotlivých částech jako u předchozí kapitoly, ale pokusím se pouze popsat některé odlišnosti.

Předchozí systém Tieto Ask neměl žádného zákazníka, který by na něj měl přesné požadavky. Vznikal jako zcela nová služba, která by mohla obohatit firmu, proto jsme měli daleko volnější prostor pro jeho vývoj. Zatímco Czech Academy jsme vyvíjeli podle požadavků zákazníka, kterým byli zaměstnanci z personálního oddělení.

Vývoj provázely pravidelné schůzky, kde jsme se jednou za tři týdny sešli se zákazníkem. Zde nám byly vysvětlovány požadované funkce systému. Na každé schůzce se pak vytvořil seznam úkolů, které musely být v daném termínu splněny. Také jsme předváděli již dokončenou funkční část systému a tím získávali zpětnou vazbu pro další vývoj. Ke správě úkolů jsme využívali plánovacího nástroje JIRA, do kterého je možné vkládat úkoly s možností nastavení kategorie, priority a dalších důležitých parametrů. U právě zpracovávaných úkolů se v tomto nástroji nastavil stav "In progress", naopak dokončeným úkolům se nastavil stav "close". Tímto způsobem jsme získali přehled nad průběhem vývoje systému.

5.1 Funkcionalita systému

Jak již bylo zmíněno, systém měl nahradit starší již používaný, proto u tvorby funkcionality nám pomohly také zdrojové kódy starší aplikace, ke kterým jsme získali přístup. Ovšem zásadní pro nás byly schůzky se zákazníkem, jelikož některé funkce v starším systému se neosvědčily a byl požadavek na jejich vylepšení nebo úplné zrušení. Základní požadovaná funkcionalita:

- Autentizace uživatelů pomocí firemního LDAP nebo z databáze
- Role v systému Admin, uživatel a lektor
- Uživateli se zobrazují tréninky a má možnost přihlášení na volný termín
- Lektor zakládá tréninky, nastavuje jejich termíny a posílá pozvánky
- Admin nastavuje role uživatelů a schvaluje nově založené tréninky
- Kalendář s přehledem probíhajících tréninků a možností přihlášení
- Hromadné posílání pozvánek a notifikací pomocí mailu
- Vyhledávání tréninků podle jména, lektora a kompetencí

5.2 Shrnutí vývoje aplikace

Při vývoji jsme pracovali se stejnými nástroji jako u předchozího systému, takže základem byl opět Spring framework, Hibernate, Maven a další. Velkou výhodou tedy bylo, že jsme mohli čerpat již nabyté znalosti.Už jsme neztráceli tolik času seznamováním se s novými nástroji, ale více se věnovali tvorbě samotné funkcionality. Vývoj systému nám zabral většinu času druhé poloviny praxe, což bylo asi 30 dní. V době psaní této práce nebyl ještě systém dokončen. Měli jsme vytvořenou funkční aplikaci, kde byla vyřešena základní funkcionalita. Ovšem stále přibývaly nové požadavky a systém se dále rozšiřoval. Už na začátku vývoje se počítalo s větší časovou náročností a bylo pravděpodobné, že kompletní systém nedokončíme. Proto byl kladen důraz na přehlednost zdrojového kódu, aby na rozšíření aplikace mohli pracovat další vývojáři po nás.

6 Použité technologie a nástroje

6.1 Java

Java je multiplatformní, objektově orientovaný programovací jazyk se silně typovou kontrolou. Byl vyvinut společností Sun Microsystems v roce 1995. V roce 2009 byla společnost Sun Microsystems odkoupena společností Oracle. Java používá jednoduchou syntaxi, která vychází z jazyka C/C++. Jedná se o takzvaný interpretovaný jazyk, který je převáděn do mezikódu (byte code). Tím je formát nezávislý na architektuře počítače a může pracovat na libovolném zařízení, které má k dispozici virtuální stroj Javy (Java Virtual Machine).

6.2 Spring Framework

Spring je open source framework pro vývoj aplikací na platformě Java. Hlavním důvodem vzniku frameworku bylo usnadnění vývoje J2EE aplikací. První verze byla zveřejněna v březnu 2014. Jde o modulární framework a umožňuje načíst jen část, která se nám v projektu hodí. Využívá návrhového vzoru Inversion of Control k odstranění těsných programových vazeb jednotlivých POJO objektů. Tento návrhový vzor je jádrem Spring Frameworku a funguje na principu přesunutí zodpovědnosti za vytvoření a provázání objektů z aplikace na framework. K získání objektů framework využívá vsazování závislostí (Dependency Injection). Dependency Injection je specialní případ Inversion of Control a řeší způsob vložení objektů. Objekty jsou většinou vytvořeny na základě konfiguračního souboru v XML, který obsahuje jejich definice.

6.3 Hibernate

Hibernate je framework vytvořený v jazyce Java určený pro objektově-relační mapování. Umožňuje zachovat stavy objektů mezi dvěma spuštěnými aplikacemi, tedy udržuje data persistentní. Využívá k tomu ORM, které mapuje Java objekty na entity relační databáze. Jedna z možností použití Hibernate je pomocí mapovacího souboru, což je XML soubor, který popisuje jak objekty namapovat. Druhá možnost je využití anotací v Java souborech nad příslušnými atributy a třídami, které se mají mapovat.Hlavní výhodou Hibernate je odstínění od specifik jednotlivých databází.

6.4 MVC

Model-View-Controller je koncept, který rozděluje prezentační vrstvu aplikací do tří nezávislých komponent. Patří mezi ně Model, View a Controller. Model specifikuje reprezentaci informací, s kterými aplikace pracuje. View převádí data do podoby vhodné k prezentaci uživateli. Controller zpracovává události nejčastěji pocházející od uživatele a zajišťuje změny v modelu a ve view.

6.5 Bootstrap

Twitter Bootstrap je volně dostupný soubor nástrojů pro vytváření uživatelského rozhraní webových aplikací. Nabízí mnoho prvků, které využívají technologie HTML, CSS, Javascript a snadno se dají implementovat do vlastní stránky.

6.6 Javascript / Jquery

Javascript je dynamický objektově orientovaný programovací jazyk. Nejčastěji se používá u webových aplikací k implementaci skriptů na straně klienta. Využívá syntaxe, která vychází z C/C++. Jquery je Javascriptová knihovna, která zjednodušuje skriptování na straně klienta. V současnosti jde o nejpoužívanější knihovnu Javascriptu.

6.7 Git

Jde o distribuovaný systém správy verzí. Vytvořil ho Linus Torvalds v roce 2005 původně pro vývoj jádra Linux. Git je free software, který v současnosti vyvíjí Junio Hamano. Git poskytuje uživateli lokální kopii historie vývoje, která se pak kopíruje na další uložište.

6.8 PostgreSQL

PostgreSQL je objektově-relační databázový systém vydávaný pod licencí MIT. Jedná se o plnohodnotný relační databázový systém vyvíjený jako open source, který obsahuje většinu SQL92 a SQL99 datových typů. Splňuje podmínky ACID, podporuje cizí klíče, JOIN operace, pohledy a jak spouštěné, tak uložené procedury.

6.9 Apache Tomcat

Apache Tomcat je open source webový server vyvíjený společností Apache Software Foundation. Tomcat implementuje několik Java EE implementací jako jsou Java Servlet, JavaServer Pages, Java EL, WebSocket a poskytuje webové prostředí pro spouštění Java aplikací. Hlavní komponenty, které tvoří Tomcat, jsou Catalina (Servlet container), Coyote (HTTP connector) a Jasper (JSP engine).

6.10 Apache Maven

Maven je nástroj pro usnadnění práce při buildování aplikací napsaných převážně v jazyce Java. Maven je postaven na modulární architektuře a obstarává dodání a spouštění jednotlivých pluginů, které jsou nakonfigurovány v souboru pom.xml. Všechny nakonfigurované pluginy a knihovny jsou automaticky staženy z internetového repozitáře do lokálního repozitáře v počítači a připojeny k projektu.

7 Závěr

7.1 Uplatněné a scházející znalosti získané během studia

Při vykonávání odborné praxe jsem využil hlavně znalosti objektově orientovaného programování v Javě, které jsem získal v předmětu Programovací jazyky I. Dále byly užitečné znalosti databází z předmětů Databázové a informační systémy a Úvod do databázových systémů. Pro obecné znalosti s vývojem informačních systému byl velmi užitečný předmět Vývoj informačních systémů, především jsem využil vědomosti pro rozdělení systému do třívrstvé architektury.

Scházely mi znalosti Spring framework, Hibernate a Maven. Tyto technologie, ale využívají Javy a XML, které mi byly známé, takže nebylo obtížné se je postupně doučit. Dále jsem neměl zkušenosti s verzovacími nástroji pro sdílení kódu, v mém případě to byl Git.

7.2 Zhodnocení dosažených výsledků

Praxi hodnotím jako pozitivní zkušenost a bylo pro mě velkým přínosem vyzkoušet si práci ve firemním prostředí. Ve společnosti Tieto se nejčastěji pracuje na rozsáhlých komerčních projektech s velkým počtem lidí v týmu. Já jsem byl spíše výjimkou a po celou dobu praxe jsem pracoval na menších projektech pouze ve dvojici s jiným studentem, za občasné odborné pomoci konzultanta. Hlavní výhodou v mém pracovním zařazení bylo, že jsem si vyzkoušel celý proces vývoje informačního systému. Nováčci ve velkých týmech si většinou vyzkouší práci pouze na jedné části systému, nejčastěji na prezentační vrstvě. Zatímco já jsem spolu s mým kolegou měl možnost vytvořit kompletní informační systém od úplného základu. Tím jsem získal zkušenosti jak s návrhem informačního systému, tak s jeho implementací na všech vrstvách třívrstvé architektury. Naučil jsem se jak správně spolupracovat s ostatními kolegy, abychom dosáhli efektivního řešení úkolů a využívat ke sdílení zdrojového kódu verzovací nástroj Git. Vyzkoušel jsem si jak probíhá komunikace se zákazníkem při vývoji informačního systému. Další velkou výhodou bylo získání zkušeností s moderními technologiemi jako je například Spring framework, který v současnosti patří mezi nejpoužívanější frameworky pro tvorbu webových aplikací na Java platformě.

8 Reference

[1] Dokumentace Spring Framework [online]. 2015 [cit. 2015-04-23]. Dostupné z WWW: <http://spring.io>.

[2] Dokumentace Hibernate [online]. 2015 [cit. 2015-04-23]. Dostupné z WWW: <http://hibernate.org/orm/documentation/>.

[3] Tieto Czech s.r.o., informace o společnosti [online]. 2015 [cit. 2015-04-23]. Dostupné z WWW: <http://www.tieto.cz/tieto-o-nas>.

[4] Wikipedie: Git [online]. 2015 [cit. 2015-04-23] Dostupné z WWW: <http://en.wikipedia.org/wiki/Git_(software)>.

[5] Wikipedie: Apache Tomcat [online]. 2015 [cit. 2015-04-23] Dostupné z WWW: <http://en.wikipedia.org/wiki/Apache_Tomcat>.

[6] Wikipedie: PostgreSQL [online]. 2015 [cit. 2015-04-23] Dostupné z WWW: <http://en.wikipedia.org/wiki/PostgreSQL>.

[7] Wikipedie: Apache Maven [online]. 2015 [cit. 2015-04-23] Dostupné z WWW: <http://en.wikipedia.org/wiki/Apache_Maven>.

[8] Wikipedie: Java [online]. 2015 [cit. 2015-04-23] Dostupné z WWW: <http://en.wikipedia.org/wiki/Java_(programming_language)>.